MATEMÁTICA APLICADA À COMPUTAÇÃO

AULA 1

Prof. Ricardo Alexandre Deckmann Zanardini

CONVERSA INICIAL

Aqui, abordaremos os principais assuntos que servirão de base para a computação. Vivemos em um mundo onde a matemática está presente diretamente ou indiretamente em tudo o que vemos ou fazemos. A origem da palavra matemática está na palavra grega *mathema*, que significa conhecimento. Inicialmente, trataremos de tópicos relacionados à lógica matemática. Em seguida, estudaremos tópicos relacionados às bases numéricas, representações de números reais, conjuntos, funções, matrizes, vetores, probabilidade e estatística, criptografia e muito mais. A cada etapa, além de estudarmos os principais assuntos, teremos exemplos, aplicações e uma introdução ao Python, uma poderosa e simples linguagem de programação. Desde já, nos colocamos à disposição para ajudarmos no que for necessário. Bons estudos!

TEMA 1 – PROPOSIÇÕES

A lógica clássica é uma ciência que preza pela organização e pela clareza do raciocínio. Durante muito tempo, a lógica foi considerada como instrumento indispensável para o pensamento científico. Atualmente, a lógica é uma parte importante no processo dedutivo das ciências. Da lógica clássica, surgiu a lógica matemática. A lógica matemática está presente em diversos ramos da ciência e tem aplicações extremamente importantes em processos dedutivos, na computação e nos problemas de inteligência artificial. Baseada em afirmações que podem ser classificadas como verdadeiras ou falsas, a lógica matemática é uma ciência útil e importante para a organização do pensamento humano. Nesta etapa, abordaremos as proposições e os operadores lógicos, elementos muito importantes na computação.

Para começarmos a tratar dos temas relacionados à lógica, precisamos falar sobre as proposições, um conjunto de palavras ou símbolos que retratam um pensamento de sentido completo e que pode ser classificado como verdadeiro ou falso.

Afirmações do tipo “4 é par”, “Paris é a capital da França” ou “4+1=10” são exemplos de proposições. É claro que as duas primeiras afirmações podem ser classificadas como verdadeiras, enquanto a última é uma afirmação falsa.

É importante ressaltar que nem toda frase ou um conjunto de símbolos quaisquer é uma proposição. As expressões: “Será que vai chover?”, “Oi!” e

“Tudo bem com você?”, por exemplo, não são proposições, pois não expressam uma afirmação de sentido completo, verdadeira ou falsa.

As proposições são objetos fundamentais para o estudo da lógica, pois a partir delas temos a possibilidade de fazer afirmações e de exprimir um juízo formado sobre determinada questão.

A lógica matemática está baseada em dois princípios fundamentais que regem todo o pensamento clássico.

I. Princípio da não contradição: uma proposição não pode ser classificada como verdadeira e falsa simultaneamente.
II. Princípio do terceiro excluído: uma proposição é verdadeira ou falsa, e não existe a possibilidade de um terceiro caso.

Logo, estes dois princípios nos motivam a determinar o conceito de *valor lógico de uma proposição*: se uma proposição é verdadeira, então seu valor lógico é a verdade (representado por V ou por 1), e se uma proposição é falsa, então seu valor lógico é a falsidade (representado por F ou por 0).

Por exemplo, o valor lógico da proposição p:”10 > 2” é a verdade (V) e o valor lógico da proposição q:”3 é um número par” é a falsidade (F). Escrevemos que V(p)=V e V(q)=F onde V(p) é o valor lógico de p e V(q) é o valor lógico de q.

Observe que as proposições simples (uma proposição p é dita simples se p não contém nenhuma outra proposição como parte integrante de si mesma) podem ser representadas por letras minúsculas p, q, r…, denominadas *letras proposicionais*. Uma ou mais proposições simples podem ser combinadas para formarem uma proposição composta. Geralmente, as proposições compostas são representadas por letras latinas maiúsculas, como P, Q, R… As seguintes proposições são exemplos de proposições compostas.

- P: Julia é médica **ou** Júlia é advogada.
- Q: Gustavo trabalha durante o dia **e** Gustavo estuda à noite.
- R: **Se** Anderson estudar, **então** terá um bom desempenho nas avaliações.
- S: **Não** é verdade que 2 é maior do que 4.
- T: Vou comprar um carro **se e somente se** receber um aumento salarial.

É fácil perceber que para obtermos as proposições compostas, não basta apenas escrever diversas proposições simples, mas sim interligá-las com termos específicos denominados *conectivos*.

Um conectivo é formado por uma ou mais palavras e utilizado para, a partir de proposições simples, formar proposições compostas.

Os conectivos mais usuais são: “e”, “ou”, “não”, “se… então…” e “…se e somente se…”.

É importante ressaltar que as proposições compostas podem ser obtidas a partir de uma ou mais proposições simples e de uma ou mais proposições compostas. Por exemplo, considerando as seguintes proposições simples:

- p: “Hoje é segunda-feira.”
- q: “Tenho que trabalhar.”
- r: “Vou ser despedido.”

Podemos formar outras proposições compostas, tais como:

- P: “Se hoje é segunda-feira, então tenho que trabalhar.”
- Q: “Tenho que trabalhar ou vou ser despedido.”

E ainda:

- R: “Se hoje é segunda-feira, então tenho que trabalhar ou vou ser despedido.”
- S: “Tenho que trabalhar ou vou ser despedido se e somente se hoje é segunda-feira.”

Sabemos que uma proposição pode assumir apenas dois valores lógicos: a verdade e a falsidade. Desta forma, é relativamente fácil determinar o valor lógico de uma proposição simples: basta decidir se a afirmação que está sendo feita é verdadeira ou falsa. Agora, se tivermos uma proposição composta, como faremos para decidir se essa afirmação é verdadeira ou falsa? Imagine uma afirmação do tipo “Fagundes é engenheiro e trabalha muito todos os dias”. Não sabemos se Fagundes é realmente engenheiro e muito menos se ele trabalha muito todos os dias. Supondo que “Fagundes é engenheiro” é uma afirmação verdadeira e que “ele trabalha muito todos os dias” também é verdadeira, então podemos imaginar que a proposição composta “Fagundes é engenheiro e trabalha muito todos os dias” também é verdadeira.

Mas se uma dessas proposições simples for falsa, o que acontece com o valor lógico da proposição composta? Ela é verdadeira ou falsa? E se as duas proposições simples forem falsas? Para decidirmos qual é o valor lógico de uma proposição composta, precisamos saber o que são operações lógicas e quais os valores lógicos são resultantes destas operações.

As operações lógicas são extremamente importantes para podermos determinar a validade ou não de proposições compostas. As operações lógicas estão relacionadas com os conectivos. As mais importantes são: Negação (“não”), Conjunção (“e”), Disjunção (“ou”), Condicional (“se… então…”) e a Bicondicional (“…se e somente se…”).

Na computação, a negação é chamada de NOT, a conjunção de AND e a disjunção de OR. Temos também NAND que é a negação da conjunção, NOR que é a negação da disjunção, e XOR, a disjunção exclusiva.

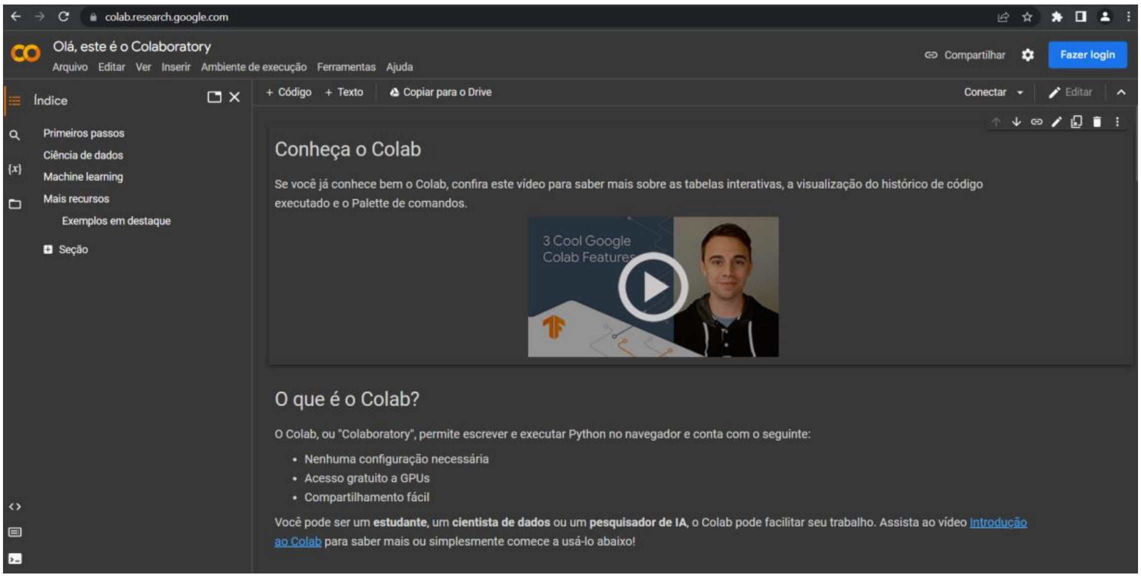
Vamos tratar detalhadamente cada uma delas e apresentar também uma introdução ao uso do Python relacionada aos operadores lógicos.

TEMA 2 – OPERADORES LÓGICOS SIMPLES (NOT, AND E OR)

Antes de tratarmos dos operadores lógicos, vamos falar um pouco a respeito do Python, uma linguagem de programação que teve início em 1991, simples e poderosa, que roda em uma grande quantidade de sistemas.

Mesmo tendo diversas opções de IDEs (*Integrated Development Environment* – Ambiente de Desenvolvimento Integrado), utilizaremos um ambiente colaborativo *on-line* denominado de *Google Colaboratory* e mais conhecido como *Google Colab*, pois possibilita o uso das funcionalidades do Python diretamente no navegador.

Para utilizarmos o Google Colab, basta ter uma conta Google e acessar o endereço colab.research.google.com.



2.1 Negação (~)

A negação é a operação lógica associada ao conectivo “não”. A negação de uma proposição p é a proposição ~p, dita “não p”, tal que o valor lógico de ~p é a verdade quando p é uma proposição falsa e ~p é falsa quando o valor lógico de p é a verdade.

Podemos representar também por p’ a negação da proposição p.

O valor lógico da negação de uma proposição é dado pela seguinte tabela-verdade:

p	p’
V	**F**
F	**V**

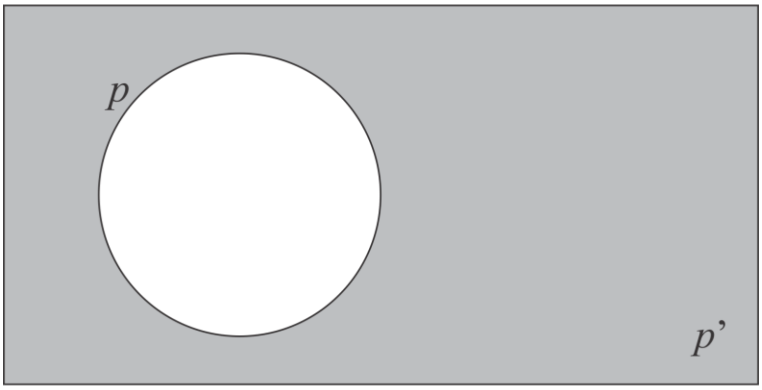
Logo, podemos dizer que a negação da verdade é a falsidade e a negação da falsidade é a verdade.

Utilizando os símbolos 0 e 1, temos:

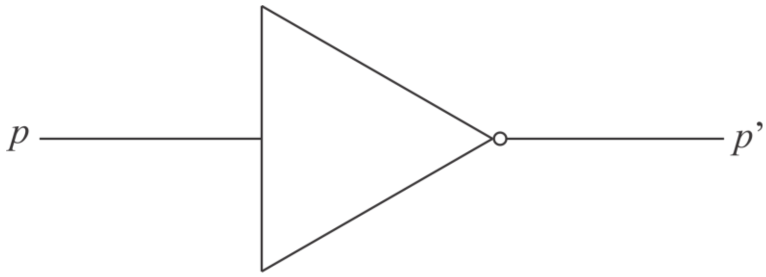
p	p’
1	**0**
0	**1**

Na tabela-verdade, temos todas as possibilidades de valores lógicos para a proposição p e os respectivos valores lógicos da proposição p’.

A negação pode ser representada por meio de um diagrama. Note que se a proposição p corresponde ao círculo, a negação é a região fora do círculo.



Também é utilizada a representação de porta lógica:



Alguns exemplos relacionados à negação:

a) A proposição p: 2+3=5 é verdadeira e a negação ~p: 2+3≠5 é uma proposição falsa. Logo, V(p)=V e V(~p)=F.
b) A proposição q: 4-2=2 é verdadeira e ~q: 4-2≠2 é falsa. Logo, V(q)=V e V(~q)=F.
c) A proposição r: 10>12 é falsa e a sua negação ~r: 10≤12 é verdadeira. Neste caso, V(r)=F e V(~r)=V.

Em linguagem corrente, podemos dizer que a negação da proposição “Gosto de estudar lógica”, por exemplo, pode ser escrita das seguintes formas: “Não gosto de estudar lógica”, “Não é verdade que gosto de estudar lógica” ou “É falso que gosto de estudar lógica”.

Podemos utilizar o Python para avaliarmos o valor lógico de expressões, bem como podemos utilizar a negação em aplicações feitas em Python.

Há muitas possibilidades e abordaremos algumas delas.

Inicialmente, vamos pensar em valores atribuídos a duas variáveis e na relação entre estes valores.

Podemos, por exemplo, considerar a variável “idadeAna” para atribuir a esta variável um valor associado à idade atual de uma pessoa e a variável “idadeBeatriz” para armazenar a idade atual de outra pessoa:

idadeAna=32

idadeBeatriz=29

Note que o valor 32 foi atribuído à variável “idadeAna” e o valor 29 foi atribuído à variável “idadeBeatriz”. Em seguida, podemos comparar estas idades utilizando, por exemplo, o sinal de < (menor que):

idadeAna<idadeBeatriz

Se digitarmos estes elementos em uma célula de código do Google Colab e clicarmos no ícone “Executar célula” ao lado da célula de código ou utilizarmos o atalho “CTRL + Enter”, o Python irá avaliar se, a partir dos valores atribuídos às variáveis “idadeAna” e “idadeBeatriz”, a expressão “idadeAna<idadeBeatriz” é verdadeira ou falsa.

```
idadeAna=32
idadeBeatriz=29
idadeAna<idadeBeatriz
```

```
False
```

O resultado obtido é “False”, ou seja, a expressão é falsa, pois a idade de Ana (32) não é menor do que a idade de Beatriz (29).

Como a negação associada à desigualdade < (menor que) é a desigualdade >= (maior ou igual a), se quisermos saber qual é o valor lógico da negação de “idadeAna<idadeBeatriz”, podemos escrever “idadeAna>=idadeBeatriz” ou então, de forma equivalente, “not(idadeAna<idadeBeatriz)”.

```
idadeAna=32
idadeBeatriz=29
idadeAna>=idadeBeatriz
```

```
True
```

```
idadeAna=32
idadeBeatriz=29
not(idadeAna<idadeBeatriz)
```

```
True
```

Como as duas formas são equivalentes, em ambas o resultado obtido é "True", ou seja, "Verdade".

No Python, podemos atribuir um valor lógico a uma variável. Os possíveis valores lógicos são "True" que representa a verdade e "False" que representa a falsidade.

Por exemplo, podemos atribuir o valor lógico verdadeiro a uma variável p e obter o respectivo valor lógico de p' fazendo:

p=True

not(p)

Como resultado, temos "False":

```
p=True
not(p)
```

```
False
```

A partir do que foi tratado até o momento, podemos criar uma sequência de comandos para que, a partir de uma nota obtida em uma disciplina, possamos saber se o estudante está aprovado ou não. Sabendo que para notas abaixo de 70, o resultado é a reprovação, se a nota obtida for maior ou igual a 70, o resultado é a aprovação, vamos criar uma sequência bem simples para fazermos isto por meio do Python.

O primeiro passo é sabermos qual foi a média obtida e atribuirmos este valor a uma variável. Podemos chamar a variável de M e na própria célula de código atribuirmos um valor para ela, por exemplo, M=80. Mas também podemos utilizar o comando "input" e digitar o valor da média em um campo específico. Como o comando "input" cria uma sequência de caracteres (string) e precisamos de um valor numérico associado a M, faremos "float(input())", ou seja, utilizaremos a função “float” para converter o número que está no formato de string para o formato de ponto flutuante (float). Assim, temos:

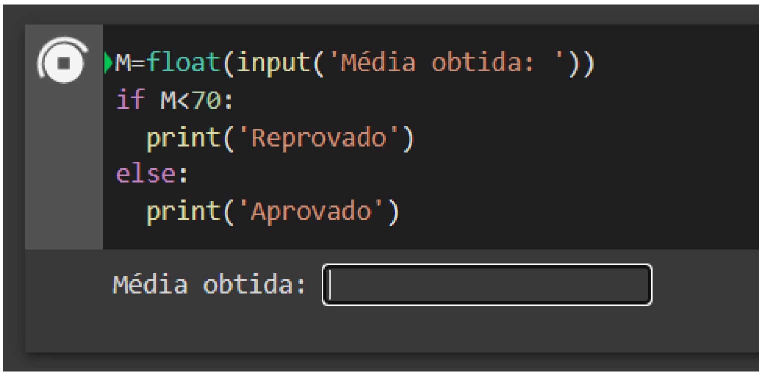
```
M=float(input('Média obtida: '))
```

Tendo o valor da média, precisamos apresentar o resultado “aprovado” ou “reprovado”. Se a média for menor do que 70 o resultado é “reprovado”. Senão, o resultado é “aprovado”. O “se” é dado por “if” e o “senão” por else. A sequência de comandos é:

```
M=float(input('Média obtida: '))
if M<70:
 print('Reprovado')
else:
 print('Aprovado')
```

Onde a função “print” apresenta o resultado escrito entre aspas.

Executando a sequência de comandos, temos inicialmente o campo para informarmos a média:



Após digitarmos o valor e apertarmos a tecla “Enter”, é apresentado o resultado:

```
Média obtida: 35.7
Reprovado
```

ou

```
Média obtida: 90
Aprovado
```

2.2 Conjunção (∧)

Um conectivo muito utilizado é o “e”. A conjunção é a operação lógica associada a este conectivo. A conjunção de duas proposições p e q, representada por p∧q ou por p.q é verdadeira sempre que p e q são verdadeiras. Nos demais casos, a conjunção é falsa.

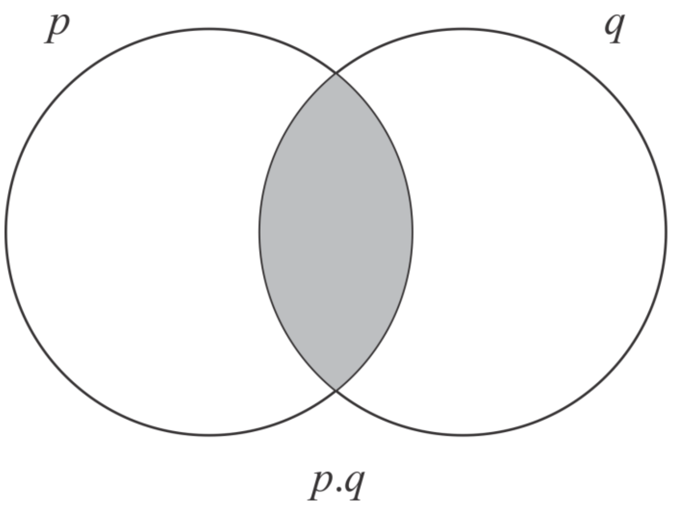
A tabela-verdade da conjunção é dada como segue:

p	.	q
V	**V**	V
V	**F**	F
F	**F**	V
F	**F**	F

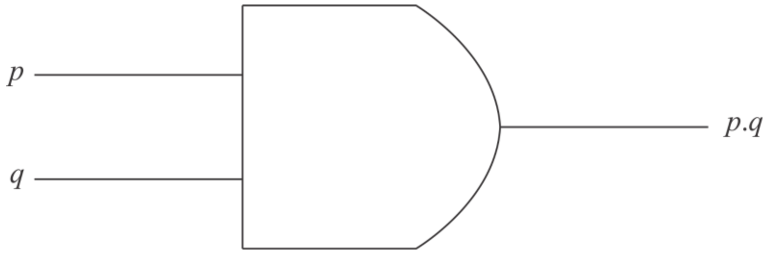
Utilizando 0 e 1 para representarmos, respectivamente, a falsidade e a verdade, temos:

p	.	q
1	**1**	1
1	**0**	0
0	**0**	1
0	**0**	0

Por meio de um diagrama, a conjunção está associada à intersecção de p e q:



Também é possível representar a conjunção por meio de portas lógicas:



A proposição composta “Gustavo trabalha durante o dia e Gustavo estuda à noite” é um exemplo de conjunção. A proposição simples “Gustavo trabalha durante o dia” está conectada com a proposição simples “Gustavo estuda à noite”, por meio do conectivo “e”. Se a proposição “Gustavo trabalha durante o dia” é verdadeira e se a proposição “Gustavo estuda à noite” é verdadeira, então a proposição composta “Gustavo trabalha durante o dia e Gustavo estuda à noite” é verdadeira. Se a proposição “Gustavo trabalha durante o dia” é verdadeira e se a proposição “Gustavo estuda à noite” é falsa, então a proposição “Gustavo trabalha durante o dia e Gustavo estuda à noite” é falsa. A proposição “Gustavo trabalha durante o dia e Gustavo estuda à noite” também é falsa quando a proposição “Gustavo trabalha durante o dia” é falsa e a proposição “Gustavo estuda à noite” é verdadeira e quando as proposições “Gustavo trabalha durante o dia” e “Gustavo estuda à noite” são falsas.

Um exemplo muito comum está associado à aprovação de um estudante em uma determinada disciplina presencial onde a média precisa ser maior ou igual a 70 e o número de faltas menor ou igual a 25% do número de aulas ministradas. Se o total de aulas corresponde a 80 horas, o máximo de faltas

(25% de 80) é 20. Podemos ter agora uma sequência de comandos com duas condições e o operador lógico “AND”:

```
M=float(input('Média obtida: '))
F=float(input('Total de faltas: '))
if M>=70 and F<=20:
 print('Aprovado')
else:
 print('Reprovado')
```

Fazendo algumas combinações de possibilidades, temos os seguintes resultados:

```
Média obtida: 89
Total de faltas: 8
Aprovado
```

```
Média obtida: 89
Total de faltas: 23
Reprovado
```

```
Média obtida: 65
Total de faltas: 10
Reprovado
```

```
Média obtida: 65
Total de faltas: 22
Reprovado
```

Para a aprovação, as duas condições precisam ser verdadeiras. Caso pelo menos uma delas seja falsa, o resultado é a reprovação.

Vamos agora pensar que temos quatro elementos para compor a média de uma disciplina: duas atividades pedagógicas on-line (APOL) com um peso de 15% (0,15) cada, uma atividade prática (AP) com peso de 40% (0,4) e uma prova objetiva (PO) com peso de 30% (0,3).

Temos também algumas possibilidades de resultados.

- Se a média for menor do que 30, o estudante está reprovado.
- Se a média for maior ou igual a 30 e menor do que 67, o estudante deverá fazer uma prova de exame final. Caso a média seja maior ou igual a 67 e

menor do que 70, é feito um arredondamento da nota e o estudante está aprovado.

- Se a média for maior ou igual a 70, o estudante está aprovado.

Em Python, temos:

```
APOL1=float(input('Nota da APOL 1: '))
APOL2=float(input('Nota da APOL 2: '))
AP=float(input('Nota da Atividade Prática: '))
PO=float(input('Nota da Prova Objetiva: '))
M=0.15*APOL1+0.15*APOL2+0.4*AP+0.3*PO
print(f'Média: {M}')
if M<30:
 print('Reprovado')
elif M>=30 and M<67:
 print('Em exame')
elif M>=67 and M<70:
 print('Aprovado por arredondamento')
else:
 print('Aprovado')
```

Para calcularmos a média, utilizamos as porcentagens nas respectivas formas decimais. Um detalhe importante é o uso do ponto (.) para separarmos as casas decimais. A soma é feita por meio do operador “+” e a multiplicação é feita por meio do operador “*”.

O termo “elif” é uma forma resumida de “else if”, ou seja, do termo “então se…” que permite que possamos colocar uma sequência de condições “se” quando necessário.

Para a apresentação da média, utilizamos uma f-string que é uma forma simples de inserirmos o valor de uma ou mais variáveis em uma frase:

```
print(f'Média: {M}')
```

Entre as chaves adicionamos a variável, neste caso, M.

Se quisermos a apresentação de um valor inteiro no final, podemos acrescentar “:.0f” após o termo “M” para indicar que teremos zero casas decimais:

```
print(f'Média: {M:.0f}')
```

Assim, temos um exemplo:

```
AP=float(input('Nota da Atividade Prática: '))
PO=float(input('Nota da Prova Objetiva: '))
M=0.15*APOL1+0.15*APOL2+0.4*AP+0.3*PO
print(f'Média: {M:.0f}')
if M<30:
 print('Reprovado')
elif M>=30 and M<67:
 print('Em exame')
elif M>=67 and M<70:
 print('Aprovado por arredondamento')
else:
 print('Aprovado')
```

```
Nota da APOL 1: 100
Nota da APOL 2: 80
Nota da Atividade Prática: 90
Nota da Prova Objetiva: 100
Média: 93
Aprovado
```

2.3 Disjunção (V)

Outra operação lógica bastante importante é a disjunção, geralmente representada por "V" ou por "+".

A disjunção de duas proposições p e q é verdadeira quando pelo menos uma das proposições é verdadeira. A disjunção é falsa quando as proposições p e q são falsas.

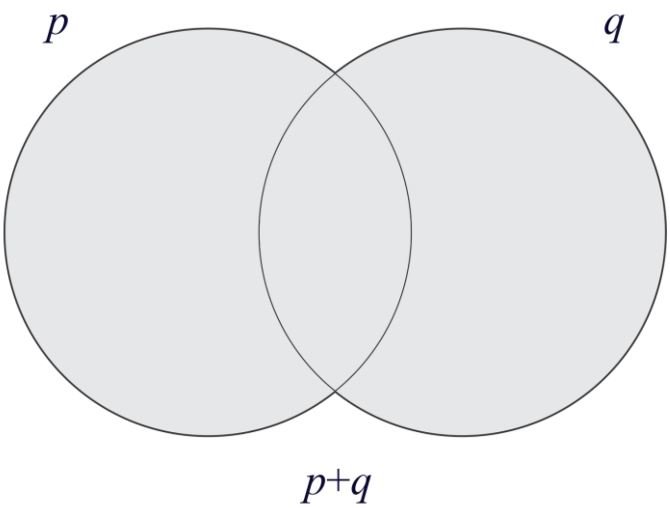
A tabela-verdade da conjunção é dada como segue:

p	+	q
V	**V**	V
V	**V**	F
F	**V**	V
F	**F**	F

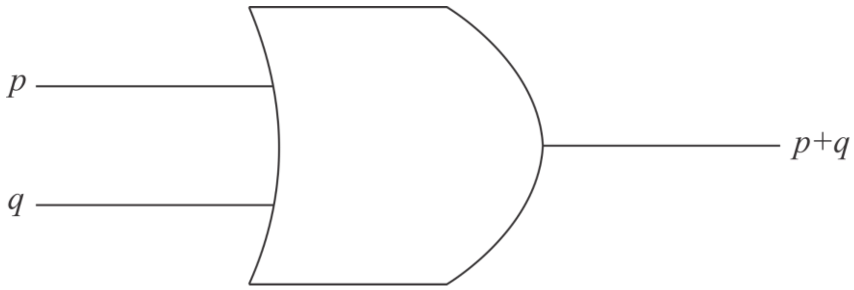
Utilizando 0 e 1 para representarmos, respectivamente, a falsidade e a verdade, temos:

p	+	q
1	**1**	1
1	**1**	0
0	**1**	1
0	**0**	0

Por meio de um diagrama, a disjunção está associada à união de p e q:



Também é possível representar a disjunção por meio de portas lógicas:



Como exemplo, podemos considerar a proposição “França é um país da Europa ou Japão é um país da América do Sul”. Se pensarmos no valor lógico da primeira proposição simples, ou seja, “França é um país da Europa”, o valor lógico é a verdade. No caso da segunda proposição simples, “Japão é um país da América do Sul”, o valor lógico é a falsidade. Mas como temos um “ou” na proposição composta “França é um país da Europa ou Japão é um país da América do Sul”, a afirmação é verdadeira, pois V ∨ F corresponde a V.

Podemos pensar em uma situação onde uma pessoa precisa apresentar um documento pessoal (RG ou CPF) para entrar em um determinado local.

Se a pessoa apresentar o RG, tem a entrada liberada. Se apresentar o CPF, tem a entrada liberada. Se apresentar RG e CPF, também tem a entrada liberada. A entrada não será liberada se não apresentar o RG e não apresentar o CPF.

Podemos escrever uma sequência simples de comandos no Python que representa esta situação:

```
rg=input('RG (s/n): ')
cpf=input('CPF (s/n): ')
if rg=='s' or cpf=='s':
 print('Entrada liberada.')
else:
 print('Entrada não autorizada.')
```

Como “n” e “s” são strings, comparamos as variáveis “rg” e “cpf” com “n” e “s” entre aspas simples.

Para diferentes combinações de entrega (s) ou não entrega (n) dos documentos, temos os seguintes resultados:

```
RG (s/n): s
CPF (s/n): s
Entrada liberada.
```

```
RG (s/n): n
CPF (s/n):
Entrada não autorizada.
```

```
RG (s/n): n
CPF (s/n): n
Entrada não autorizada.
```

TEMA 3 – OPERADORES LÓGICOS INTERMEDIÁRIOS (NAND, NOR E XOR)

Além da negação, conjunção e disjunção, podemos utilizar a negação da conjunção (NAND), a negação da disjunção (NOR) e a disjunção exclusiva (XOR).

A negação da conjunção, conhecida como NAND, tem a seguinte tabela verdade com os resultados em negrito:

~	(p	.	q)
F	V	V	V
V	V	F	F
V	F	F	V
V	F	F	F

Utilizando 0 e 1 para representarmos, respectivamente, a falsidade e a verdade, temos:

~	(p	.	q)
0	1	1	1
1	1	0	0
1	0	0	1
1	0	0	0

Note que a negação inverte todos os valores lógicos da expressão p.q, ou seja, a negação da conjunção é falsa quando os valores lógicos de p e q são verdadeiros e é verdadeira nos demais casos.

De forma equivalente, a negação de p.q também pode ser escrita como ~pv~q, pois ambas possuem tabelas-verdades com o mesmo resultado final.

Podemos pensar, por exemplo, em uma instituição financeira que concede uma modalidade de financiamento para pessoas que possuem renda mensal que pode variar de R$ 1.200,00 a R$ 3.000,00.

Assim, o empréstimo é concedido se a renda mensal está no intervalo [1200, 3000] e não é concedido se a renda mensal é menor do que R$ 1.200,00 ou maior do que R$ 3.000,00.

Outra importante operação lógica é a negação da disjunção, conhecida como NOR, cuja tabela-verdade é dada por

~	(p	v	q)
F	V	V	V

F	V	V	F
F	F	V	V
V	F	F	F

Utilizando 0 e 1, temos:

~	(p	v	q)
0	1	1	1
0	1	1	0
0	0	1	1
1	0	0	0

Logo, a negação da disjunção é verdadeira apenas quando p e q são proposições falsas. Caso contrário, a negação da disjunção é falsa.

De forma equivalente, ~(p∨q) pode ser escrita como ~p.~q.

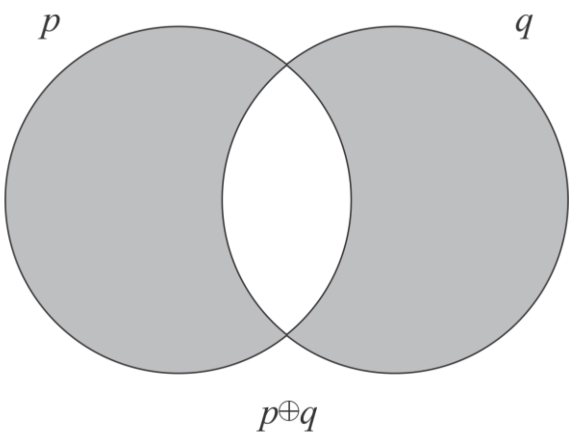
A disjunção exclusiva, conhecida como XOR, geralmente representada por $\oplus$, é verdadeira apenas quando p e q possuem valores lógicos diferentes:

p	$\oplus$	q
V	**F**	V
V	**V**	F
F	**V**	V
F	**F**	F

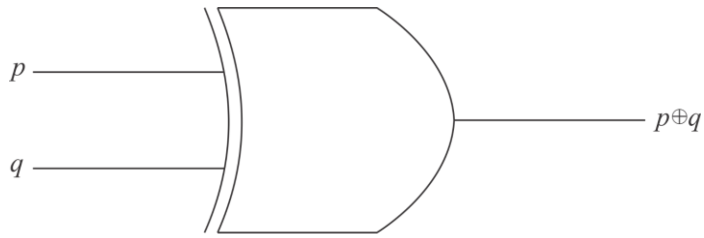
Utilizando 0 e 1 para representarmos, respectivamente, a falsidade e a verdade, temos:

p	$\oplus$	q
1	**0**	1
1	**1**	0
0	**1**	1
0	**0**	0

Graficamente, temos:



A representação por meio de portas lógicas é:



No caso da disjunção exclusiva, para que a proposição p⊕q seja verdadeira, não podemos ter p verdadeira e q verdadeira simultaneamente. Por exemplo, se considerarmos a afirmação “João é gaúcho ou pernambucano”, ela só será verdadeira se João for gaúcho ou se João for pernambucano, mas considerando que João não pode ser gaúcho e ser pernambucano ao mesmo tempo.

TEMA 4 – OPERADORES LÓGICOS AVANÇADOS (CONDICIONAL E BICONDICIONAL)

Vamos agora abordar dois operadores lógicos muito importantes em diversas situações. O primeiro deles é a condicional.

4.1 Condicional (→)

A condicional, representada por →, é uma operação lógica associada aos termos “se…, então…” e tem a seguinte tabela-verdade:

p	→	q
1	**1**	1
1	**0**	0

0	**1**	1
0	**1**	0

O valor lógico da condicional é a falsidade quando p é uma proposição verdadeira e q uma proposição falsa. Nos demais casos, a condicional é verdadeira.

Para compreendermos o significado da condicional e onde podemos utilizar esta operação lógica, vamos pensar em um exemplo.

Podemos imaginar que em uma empresa foi prometido um prêmio de R$ 100,00 para cada trabalhador caso as vendas no mês de dezembro gerassem um lucro líquido de mais de R$ 20.000,00 para a empresa.

Sendo assim, temos as seguintes proposições:

p: O lucro líquido ficou acima de R$ 20.000,00

q: Cada trabalhador recebeu R$ 100,00

Como a promessa é:

"Se o lucro líquido ficou acima de R$ 20.000,00, então cada trabalhador recebeu R$ 100,00".

Temos um exemplo de condicional.

Analisando as possibilidades, temos:

Se o lucro líquido ficou acima de R$ 20.000,00 e cada trabalhador recebeu R$ 100,00, a promessa foi cumprida e a condicional é verdadeira.

1. Se o lucro líquido ficou acima de R$ 20.000,00 e cada trabalhador não recebeu R$ 100,00, a promessa não foi cumprida e a condicional é falsa.
2. Se o lucro líquido não ficou acima de R$ 20.000,00, mas mesmo assim cada trabalhador recebeu um prêmio de R$ 100,00, a promessa não foi descumprida e a condicional é verdadeira.
3. Se o lucro líquido não ficou acima de R$ 20.000,00 e cada trabalhador não recebeu R$ 100,00, a promessa não foi descumprida e a condicional é verdadeira.

Fica fácil perceber que a única possibilidade de termos o valor lógico falso para a condicional quando a empresa atinge o objetivo e não paga o prêmio aos trabalhadores. Nos demais casos a condicional é verdadeira.

4.2 Bicondicional (↔)

A bicondicional, representada por ↔, é uma operação lógica associada aos termos "…se e somente se…" cuja tabela-verdade corresponde a:

p	↔	q
1	**1**	1
1	**0**	0
0	**0**	1
0	**1**	0

A bicondicional é verdadeira quando as duas proposições possuem o mesmo valor lógico e é falsa quando as duas proposições possuem valores lógicos diferentes.

A bicondicional difere da condicional quando p é falsa e q é verdadeira.

Pensando no exemplo da empresa que prometeu um prêmio de R$ 100,00 para cada trabalhador caso as vendas no mês de dezembro gerassem um lucro líquido de mais de R$ 20.000,00 para a empresa, se for feito o uso da bicondicional no lugar da condicional, o prêmio não poderá ser pago se o lucro líquido não for maior do que R$ 20.000,00.

Em linguagem corrente, temos agora a seguinte proposição associada à bicondicional:

"Cada trabalhador receberá R$ 100,00 se e somente se o lucro líquido da empresa ficar acima de R$ 20.000,00".

Analisando as possibilidades, temos:

1. Se o lucro líquido ficou acima de R$ 20.000,00 e cada trabalhador recebeu R$ 100,00, a promessa foi cumprida e a bicondicional é verdadeira.
2. Se o lucro líquido ficou acima de R$ 20.000,00 e cada trabalhador não recebeu R$ 100,00, a promessa não foi cumprida e a bicondicional é falsa.
3. Se o lucro líquido não ficou acima de R$ 20.000,00, mas mesmo assim cada trabalhador recebeu um prêmio de R$ 100,00, a empresa efetuou um pagamento que não estava autorizado e a bicondicional é falsa.
4. Se o lucro líquido não ficou acima de R$ 20.000,00 e cada trabalhador não recebeu R$ 100,00, a promessa não foi descumprida e a

bicondicional é verdadeira.

TEMA 5 – PROPRIEDADES

Com o que estudamos até aqui, podemos destacar algumas propriedades importantes das operações sobre proposições.

Uma delas é a dupla negação, ou seja, a negação de uma negação é equivalente à proposição original:

$$(p')' \Leftrightarrow p$$

O símbolo “⇔” indica a equivalência, ou seja, que as expressões (p’)’ e p possuem o mesmo valor lógico.

Temos também as leis idempotentes:

$$p+p \Leftrightarrow p$$

e

$$p.p \Leftrightarrow p$$

A comutatividade é válida para a conjunção e para a disjunção:

$$p+q \Leftrightarrow q+p$$

e

$$p.q \Leftrightarrow q.p$$

As leis associativas

$$(p+q)+r \Leftrightarrow p+(q+r)$$

e

$$(p.q).r \Leftrightarrow p.(q.r)$$

também são válidas.

São válidas as leis distributivas

$$p.(q+r) \Leftrightarrow (p.q)+(p.r)$$

e

$$p+(q.r)\Leftrightarrow(p+q).(p+r)$$

Por fim, temos as leis de De Morgan, que tratam da negação da conjunção e da negação da disjunção:

$$(p+q)'\Leftrightarrow p'.q'$$

e

$$(p.q)'\Leftrightarrow p' \vee q'$$

No que se refere à condicional, a expressão $q'\rightarrow p'$, denominada de *contrapositiva*, é equivalente à expressão $p\rightarrow q$:

$$p\rightarrow q\Leftrightarrow q'\rightarrow p'$$

Por exemplo, se temos as proposições simples

p: João aprendeu o conteúdo

q: João foi aprovado na disciplina

dizer que "Se João aprendeu o conteúdo, então João foi aprovado na disciplina" é equivalente a dizer que "Se João não foi aprovado na disciplina, então João não aprendeu o conteúdo".

FINALIZANDO

Estamos chegando ao final da nossa etapa sobre matemática aplicada à computação. Nesta etapa, estudamos importantes assuntos relacionados à lógica matemática, tais como proposições, operações lógicas sobre proposições e as respectivas propriedades. Também aprendemos a utilizar de forma simples o Python para resolvermos problemas relacionados a algumas das operações lógicas.

REFERÊNCIAS

CASTANHEIRA, N. P.; LEITE A. E. **Raciocínio lógico e lógica quantitativa**. Curitiba: InterSaberes, 2017.

DAGHLIAN, J. **Lógica e álgebra de Boole**. 4. ed. São Paulo: Atlas, 2012.

LEITE, Á. E.; CASTANHEIRA, N. P. **Teoria dos números e teoria dos conjuntos**. 1. ed. Curitiba: InterSaberes, 2014.

LIMA, D. M. de; GONZALEZ L. E. F. **Matemática aplicada à informática**. Porto Alegre: Bookman, 2015.

MACEDO, L. R. D. de; CASTANHEIRA, N. P.; ROCHA, A. **Tópicos de matemática aplicada**. Curitiba: InterSaberes, 2013.

NAVIDI, W. **Probabilidade e Estatística para Ciências Exatas**. Grupo A, 2012.

SILVA, F. S. C.; FINGER, M.; MEL, A. C. V. **Lógica para computação**. 2. ed. São Paulo: Cengage, 2018.

STALLINGS, W. **Criptografia e segurança de redes**: princípios e práticas. 6. ed. São Paulo: Pearson, 2015.

STEIN, C.; DRYSDALE, R.; BOGART, K. **Matemática discreta para ciência da computação**. Pearson Education do Brasil, 2013.